Combustion and Flame 174 (2016) 1–15

Contents lists available at ScienceDirect

Combustion and Flame

journal homepage: www.elsevier.com/locate/combustflame

On the merits of extrapolation-based stiff ODE solvers for combustion CFD

A. Imren*, D.C. Haworth

Research Building East, Department of Mechanical and Nuclear Engineering, The Pennsylvania State University, University Park, PA 16802, USA

ARTICLE INFO

Article history:
Received 6 May 2016
Revised 27 June 2016
Accepted 15 September 2016
Available online 28 September 2016

Keywords:
Computational fluid dynamics
Detailed chemistry
Stiff ODE solver
Dynamic adaptive chemistry
Compression-ignition engine

ABSTRACT

In applications including compression-ignition engines, there is a need to accommodate more realistic chemistry in computational fluid dynamics (CFD) simulations. Here, we consider approaches where a chemical mechanism is implemented in an application CFD code, an operator-splitting strategy is used to isolate the chemical source terms, and a stiff ordinary differential equation (ODE) solver is used to compute the changes in composition due to chemical reactions for each computational element. Chemical source terms often dominate the computational effort, and reducing the high computational cost associated with realistic chemistry has been the subject of extensive research. This includes work on improved stiff ODE solvers, which in most cases has centered on backward differentiation formula (BDF) methods. Here a different class of solvers is considered, based on extrapolation methods. Key elements of stiff ODE solvers are reviewed briefly, focusing on differences between BDF methods and extrapolation methods. Issues related to using a stiff ODE solver with operator splitting in a CFD code (where the solver is repeatedly stopped and restarted) are emphasized. Homogeneous-reactor results are presented first. There the relationship between user-specified error tolerances and solution accuracy is explored, tradeoffs between accuracy and CPU time are shown, and close-to-linear increase in CPU time with increasing chemical mechanism size is demonstrated. Engine results are presented next, including both homogeneous-charge compression-ignition engines, and direct-injection (nonhomogeneous) compression-ignition engines. There some results are presented where the stiff ODE solver is combined with a dynamic adaptive chemistry scheme. In all cases, it is found that the extrapolation solver offers significant advantages in accuracy and computational efficiency compared to the BDF solver. While the results presented are for one BDF solver (CVODE) and one extrapolation solver (SEULEX), it is anticipated that the insight into how stiff ODE solvers behave in combustion CFD will be broadly applicable to other solvers in these general classes.

1. Introduction

Across multiple domains of application, there is a pressing need to incorporate more realistic chemical kinetics in computational fluid dynamics (CFD) simulations of chemically reacting flows. For example, increasingly large chemical mechanisms are needed to accurately predict autoignition, heat release and pollutant emissions in multidimensional modeling of in-cylinder processes in compression-ignition engines, especially for the thermochemical environments that are of interest for next-generation engines [1,2]. Here, the focus is on approaches where the chemical mechanism of interest is carried directly in the CFD simulation (versus being pretabulated in flame libraries, for example), an operator-splitting strategy is invoked to isolate the chemical source terms in the governing equations, and the changes in composition due to chemical reactions for each computational element (e.g., each finite-volume cell in the case of Eulerian CFD, or each computational particle in the case of a Lagrangian-particle-based transported probability density function – PDF – method [3]) are computed by solving a system of ordinary differential equations (ODEs), typically using a stiff ODE solver. This does not preclude initial prereduction of a detailed chemical mechanism. Further information on why this approach is of interest is provided in Section 2.

In this scenario, calculation of chemical source terms usually dominates the computational effort, and several strategies have been developed to reduce the high computational overhead of carrying realistic chemistry. These include: on-the-fly mechanism reduction strategies, where the chemical mechanism is reduced locally for each computational element based on local thermochemical conditions (e.g., directed-relation-graph-based dynamic

* Corresponding author.
E-mail address: aui14@psu.edu (A. Imren).

http://dx.doi.org/10.1016/j.combustflame.2016.09.018
0010-2180/

adaptive chemistry – DRG/DAC [4,5]); storage-retrieval (tabulation) strategies, where chemical source terms are stored as they are computed and are reused later in the simulation when similar thermochemical conditions are encountered (e.g., *in situ* adaptive tabulation – ISAT [6]); clustering strategies, where computational elements having similar initial conditions are combined to reduce the number of ODE integrations required (e.g., chemistry coordinate mapping – CCM [7]); hardware-based strategies that take advantage of specific computer architectures (e.g., graphics processing units – GPUs [8]); and combinations of two or more of these strategies (e.g., combining dynamic adaptive chemistry with tabulation [9,10], or dimension reduction with tabulation [11,12]).

At some point, all of these on-the-fly chemistry acceleration strategies (as well as most prereduction strategies) require a stiff ODE solver, and improved stiff ODE solvers, tailored to the combustion chemistry problem, have been the subject of recent investigations [13–15]. Much of the work to date on stiff ODE solvers for combustion chemistry has focused on solver performance for isolated homogeneous reactors, and has not directly considered the implications on solver performance of repeated solver stopping/reinitialization in a CFD code. Also, most work has focused on backward differentiation formula (BDF) methods for solving systems of stiff ODEs. Here, a fresh look is taken at stiff ODE solvers for combustion CFD by explicitly considering the reinitialization problem, and this leads to consideration of a different class of stiff ODE solvers based on extrapolation methods [16,17].

The remainder of the paper is organized as follows. In Section 2, the motivation for operator splitting and stiff ODE solvers in combustion CFD is presented, key elements of stiff ODE solvers are discussed, the reinitialization problem is introduced, a basic DAC method is reviewed (for subsequent application to engine combustion CFD, in combination with the stiff ODE solvers), and other elements of the CFD algorithms that have been used here are described. The chemical mechanisms that have been considered are introduced in Section 3, and results are presented in Section 4. Three general configurations are considered: homogeneous reactors, homogeneous-charge compression-ignition (HCCI) engines and direct-injection compression-ignition engines. The emphasis is on comparisons of computational time required versus solution accuracy for different stiff ODE solver variants. Conclusions and next steps are summarized in the final section.

2. Operator splitting and stiff ODE solvers for combustion CFD

For the class of combustion CFD problems that is of interest here, key elements of the computational strategy are: (1) invoke operator splitting to isolate the chemical source terms in the governing equations; (2) integrate a system of highly nonlinear ODEs to compute the changes in composition due to chemical reactions over a specified computational time step using a stiff ODE solver; and (3) implement other chemistry acceleration strategies (in addition to improved stiff ODE solvers) to further reduce the computational time required for chemistry calculations – here DRG–based DAC is considered, as an example. Each of these is discussed in the following subsections. The importance of solver reinitialization is emphasized. In the final subsection, the specific CFD algorithms that have been used are described briefly.

2.1. Operator splitting

We consider a reacting mixture of N_S chemical species. The mixture composition can be expressed in terms of the species mass fractions (Y_α for species α), and the equations governing the evolution of Y_α can be written as Eulerian partial differential equations (in the case of finite-volume-based CFD, for example) or as Lagrangian ordinary differential equations (in a particle-based transported PDF method [3], for example). In either case, one can invoke operator splitting to isolate the changes in composition due to chemical reactions. Then for each computational element (Eulerian finite-volume cell, or Lagrangian particle), the change in composition due to chemical reactions over a computational time step Δt_{CFD} can be computed as the solution to a nonlinear initial-value problem:

$$\frac{d\mathbf{Y}}{dt} = \mathbf{S}\big(\mathbf{Y}(t)\big) \quad \text{with } \mathbf{Y}(t=t_0) = \mathbf{Y}_0 \ ,$$

$$\text{or} \quad \mathbf{Y}(t_0 + \Delta t_{\mathrm{CFD}}) = \mathbf{Y}_0 + \int_{t_0}^{t_0+\Delta t_{\mathrm{CFD}}} \mathbf{S}\big(\mathbf{Y}(t)\big)dt \ , \tag{1}$$

where $\mathbf{Y}$ and $\mathbf{S}$ are vectors of length $N_\phi = N_S + 1$ (N_S species mass fractions plus temperature), and $\mathbf{S}$ is the mass-based rate of production of species α or of temperature. The method that is chosen to solve Eq. 1 depends on the physical time scales in the problem, compared to the computational time step Δt_{CFD}. For typical hydrocarbon-air chemical mechanisms at engine-relevant conditions, the chemistry is *stiff*: that is, the range of relevant time scales (which can be quantified formally using eigenvalue analysis, or estimated using various approximations) varies over several orders of magnitude. The longest chemical time scale of interest is the time to reach chemical equilibrium from the prescribed initial condition: e.g., $\tau_{\mathrm{chem,max}} = \tau_{\mathrm{equil}} \approx 10^{-4} - 10^{-3}$ s. The value of $\tau_{\mathrm{chem,\ max}}$ can be even larger than this, depending on the chemical mechanism, thermochemical conditions, and the physics of interest (the time required for NO to reach equilibrium can be much longer, for example). The shortest chemical time scales are typically several orders of magnitude smaller: e.g., $\tau_{\mathrm{chem,min}} \approx 10^{-9} - 10^{-8}$ s. The value of $\tau_{\mathrm{chem,\ min}}$ can be even smaller than this, depending on the chemical mechanism and thermochemical conditions. The computational time step that is used in CFD is usually determined based on considerations other than chemistry (e.g., capturing transport or fuel-spray processes, or controlling splitting errors), and a typical value for compression-ignition engine applications is on the order of microseconds: $\Delta t_{\mathrm{CFD}} \approx 10^{-6} - 10^{-5}$ s, say. This is still orders of magnitude larger than the smallest relevant chemical time scale, and in that case, an implicit stiff ODE solver is appropriate to solve the system of ODEs given in Eq. 1.

2.2. Stiff ODE solvers

The nonlinear initial-value problem of Eq. 1 can be solved using variants of Newton iteration, where a succession of linear problems is solved whose solution eventually converges to the solution of the nonlinear problem. This involves subdividing Δt_{CFD} into smaller subintervals Δt_{ODE}. For an explicit solver, or for an implicit solver that reverts to explicit time stepping at some point, the smallest value of Δt_{ODE} can be expected to be of the order of $\tau_{\mathrm{chem,\ min}}$. To advance the solution over a subinterval of the CFD time step from time t_n (where $\mathbf{Y}_n = \mathbf{Y}(t=t_n)$ is known) to time t_{n+1} (where $\mathbf{Y}_{n+1} = \mathbf{Y}(t=t_{n+1})$ is to be found) and with $\Delta t_{\mathrm{ODE}} \equiv t_{n+1} - t_n$, the problem can be linearized as follows:

$$\begin{aligned}\mathbf{Y}_{n+1} &= \mathbf{Y}_n + \int_{t_n}^{t_{n+1}} \Big(\mathbf{S}_n + \mathbf{J}_n\big(\mathbf{Y}_{n+1} - \mathbf{Y}_n\big) + \mathcal{O}(\Delta t_{\mathrm{ODE}}^2)\Big)dt \\ &= \mathbf{Y}_n + \mathbf{S}_n \Delta t_{\mathrm{ODE}} + \mathbf{J}_n\big(\mathbf{Y}_{n+1} - \mathbf{Y}_n\big)\Delta t_{\mathrm{ODE}} + \mathcal{O}(\Delta t_{\mathrm{ODE}}^2) \ ,\end{aligned}$$

so that on rearranging and neglecting higher-order terms ($\mathcal{O}(\Delta t_{\mathrm{ODE}}^2)$):

$$\big(\mathbf{I} - \mathbf{J}_n \Delta t_{\mathrm{ODE}}\big)\big(\mathbf{Y}_{n+1} - \mathbf{Y}_n\big) = \mathbf{S}_n \Delta t_{\mathrm{ODE}} \ . \tag{2}$$

Here, $\mathbf{I}$ is the identity matrix and $\mathbf{J}_n \equiv \frac{d\mathbf{S}}{d\mathbf{Y}}|_n$ is the Jacobian matrix evaluated at time level n. The final equation provides the basis of an iterative procedure that can be used to solve for $\mathbf{Y}_{n+1}$. On each iteration, $\mathbf{S}$ is recomputed based on the most recent $\mathbf{Y}$, $\mathbf{J}$ is updated